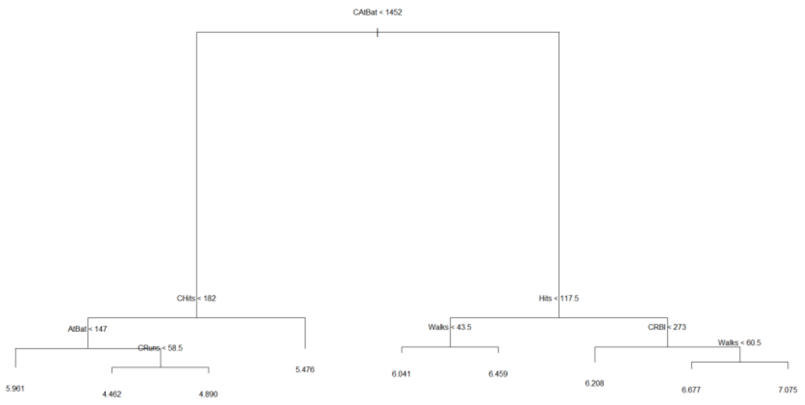
STAT 4360 (Introduction to Statistical Learning, Fall 2022)
Mini Project 5
Name: Haeun Kim

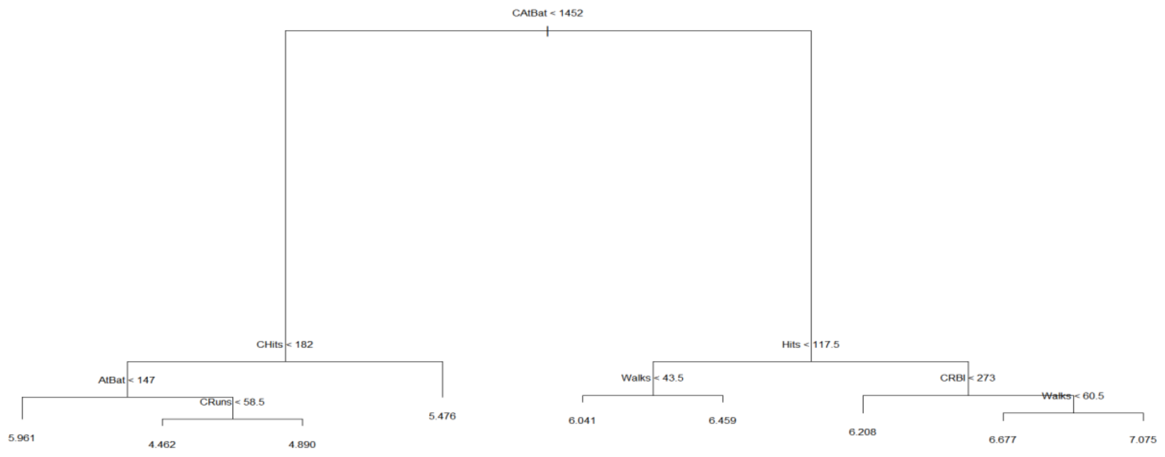
1) (a)



The test MSE of this tree by LOOCV is 0.2545162
R1=X |CatBat < 1452, CHits < 182, AtBat < 147
R2=X |CatBat < 1452, CHits < 182, AtBat >= 147, CRuns < 58.5
R3=X |CatBat < 1452, CHits < 182, AtBat >= 147, CRuns >= 58.5
R4=X |CatBat < 1452, CHits >= 182
R5=X |CatBat >= 1452, Hits < 117.5, Walks < 43.5
R6=X |CatBat >= 1452, Hits < 117.5, Walks >= 43.5
R7=X |CatBat >= 1452, Hits >= 117.5, CRBI < 274
R8=X |CatBat >= 1452, Hits >= 117.5, CRBI >= 274, Walks < 60.5
R9=X |CatBat >= 1452, Hits >= 117.5, CRBI >= 274, Walks >= 60.5

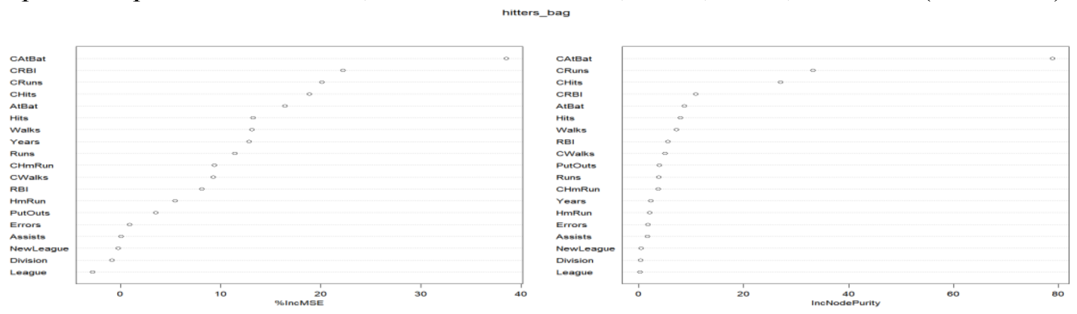
(b)

- I used LOOCV to determine whether pruning is helpful and determine the optimal size for the pruned tree. I could get the 9 as the optimal size, and performed tree to get the best pruned tree, but the best pruned tree and un-pruned tree are same. I got the estimated test MSE as 0.2574206. The most importatn predictor is CatBat.



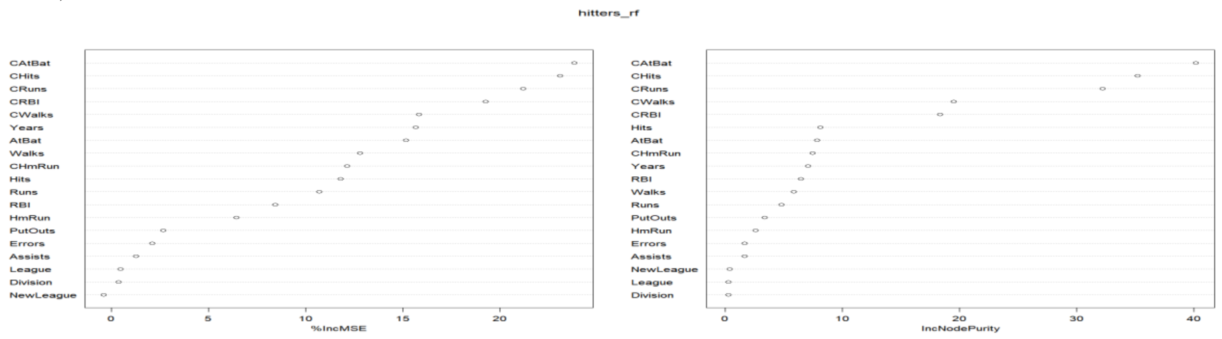
(c)

- I performed the bagging approach to analyze the data with B = 1000, and I could get the results below. The test MSE by using LOOCV is 0.1864304. According to the summury of bagging, the most importatant predictor is CAtBat, and after that CRuns, CHits, CRBI, and so on (like below).



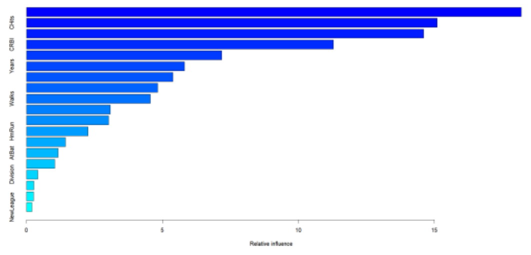
(d)

- I performed the random forest approach to analyze the data with B = 1000 and m = p/3, and I could get the results below. The test MSE by using LOOCV is 0. 0.1796064. According to the summury of random forest, the most importatant predictor is CAtBat, and after that CHits, CRuns, CWalks, and so on (like below).



(e)

- I performed the boosting approach to analyze the data with B = 1000, d = 1, and lambda = 0.01, and I could get the results below. The test MSE by using LOOCV is 0.218629. According to the summury of boosting approach, the most importatant predictor is CAtBat, and after that Chits, Cruns, CRBI, and so on (like below).



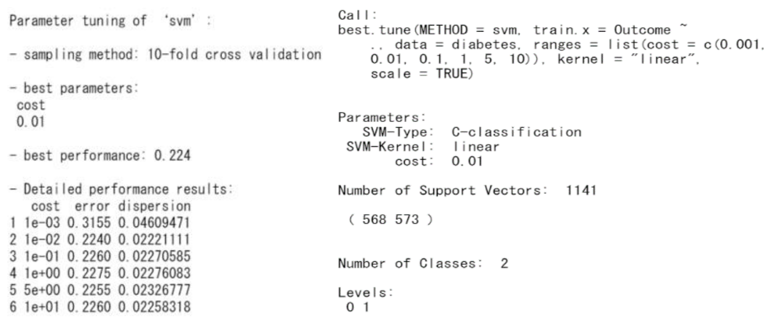
```
                 var     rel.inf
CAtBat        CAtBat 18.1973347
CHits          CHits 15.1036109
CRuns          CRuns 14.6013490
CRBI            CRBI 11.2837927
CWalks        CWalks  7.1811925
Years          Years  5.8040493
CHmRun        CHmRun  5.3791307
Hits            Hits  4.8164244
Walks          Walks  4.5524982
PutOuts      PutOuts  3.0752376
RBI              RBI  3.0178697
HmRun          HmRun  2.2563213
Runs            Runs  1.4251793
AtBat          AtBat  1.1546267
Errors        Errors  1.0347660
Division    Division  0.4070487
Assists      Assists  0.2615808
League        League  0.2555378
NewLeague NewLeague  0.1924498
```

(f)

- I would recommend the random forest because its test MSE is the lowest among other methods

2 (a)

- With the diabetes dataset (Outcome as binary response, and the 8 vaeiables as predictors), I performed to fit a support vector classifier to the data with cost parameter (0.001, 0.01, 0.1, 1, 5, 10) with 10-fold cross validation (its cross-validation default is 10-fold CV). For the best model, its cost is 0.01 and its number of support vectors is 1141. In addition, its test error rate is 0.224

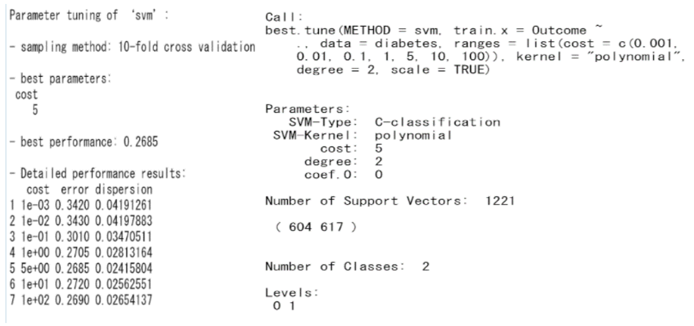


```
Parameter tuning of  'svm' :

- sampling method: 10-fold cross validation

- best parameters:
 cost
 0.01

- best performance: 0.224

- Detailed performance results:
   cost  error dispersion
1 1e-03 0.3155 0.04609471
2 1e-02 0.2240 0.02221111
3 1e-01 0.2260 0.02270585
4 1e+00 0.2275 0.02276083
5 5e+00 0.2255 0.02326777
6 1e+01 0.2260 0.02258318
```

```
Call:
best.tune(METHOD = svm, train.x = Outcome ~
    ., data = diabetes, ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10)), kernel = "linear",
    scale = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01

Number of Support Vectors:  1141

 ( 568 573 )


Number of Classes:  2

Levels:
 0 1
```

(b)

- I performed to fit a support vector machine with a polynomial kernel of degree two and cost parameter (0.001, 0.01, 0.1, 1, 5, 10, 100) with 10-fold cross validation (its cross-validation default is 10-fold CV). For the best model, its cost is 5, and its number of support vectors is 1221. In addition, its test error rate is 0.2685

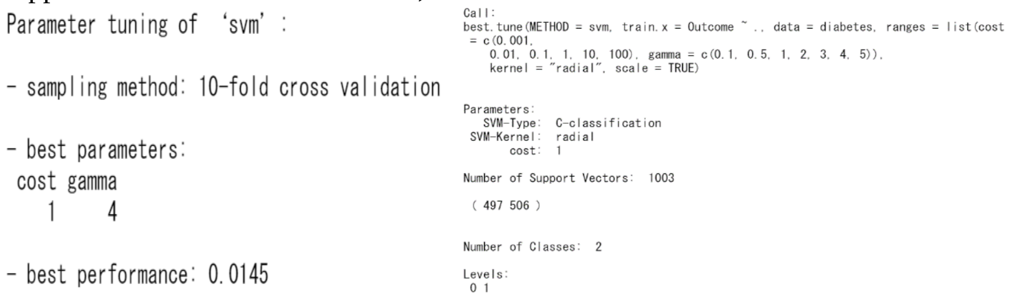


```
Parameter tuning of  'svm' :

- sampling method: 10-fold cross validation

- best parameters:
 cost
    5

- best performance: 0.2685

- Detailed performance results:
   cost  error dispersion
1 1e-03 0.3420 0.04191261
2 1e-02 0.3430 0.04197883
3 1e-01 0.3010 0.03470511
4 1e+00 0.2705 0.02813164
5 5e+00 0.2685 0.02415804
6 1e+01 0.2720 0.02562551
7 1e+02 0.2690 0.02654137
```

```
Call:
best.tune(METHOD = svm, train.x = Outcome ~
    ., data = diabetes, ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 100)), kernel = "polynomial",
    degree = 2, scale = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  5
     degree:  2
     coef.0:  0

Number of Support Vectors:  1221

 ( 604 617 )


Number of Classes:  2

Levels:
 0 1
```

(c)

- I performed to fit a support vector machine with a radial kernel with both gamma (0.1, 0.5, 1, 2, 3, 4, 5) and the cost parameter (0.001, 0.01, 0.1, 1, 10, 100) with 10-fold cross validation (its cross-validation default is 10-fold CV). For the best model, its cost is 1 and the gamma is 4, and its number of support vectors is 1003. In addition, its test error rate is 0.0145.



```
Parameter tuning of  'svm' :

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1     4

- best performance: 0.0145
```

```
Call:
best.tune(METHOD = svm, train.x = Outcome ~ ., data = diabetes, ranges = list(cost
 = c(0.001,
    0.01, 0.1, 1, 10, 100), gamma = c(0.1, 0.5, 1, 2, 3, 4, 5)),
    kernel = "radial", scale = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  1003

 ( 497 506 )


Number of Classes:  2

Levels:
 0 1
```

(d)

- The test error rate is that "support vector classifier → 0.224", "support vector machine with polynomial kernel → 0.2685", and "support vector machine with radial kernel → 0.0145" From the project 4, the test error rate is that "Linear regression → 0.2214807", "Best-subset&Forward stepwise&Backward stepwise→ 0.2199697", "Ridge regression → 0.221", and "Lasso regression → 0.2255". From the project 3, the test error rate is that "Logistic regression with LOOCV → 0.2185", "LDA → 0.223", "QDA → 0.2445", and "KKN→ 0.0015". I would recommend KKN and support vector machine with radial kernel because those test error rates are low. However, when K = 1 from the KNN, it could be overfitted, but from the project3, I got the optimal K = 1, so there could be overfitting. Therefore, considering both accuracy and safety to analysis the data, I could recommend the Support Vector Machine with radial kernel

R Code

```
library(tree)
library(ISLR2)
library(randomForest)
library(gbm)
library(e1071)
setwd("C:/Users/haeun/OneDrive/문서/STAT33550")
#Bringing the wine dataset
hitters <- read.csv("Hitters.csv")
hitters <- na.omit(hitters)
#Deleting the X variables
hitters$X <- NULL
hitters$League = as.factor(hitters$League)
hitters$Division = as.factor(hitters$Division)
hitters$NewLeague = as.factor(hitters$NewLeague)
totpred <- ncol(hitters) - 1
#Taking the log for the salary
hitters$Salary = log(hitters$Salary)

#Question 1-(a)
#Setting seed
set.seed(1)
#Making the tree
hitters.tree <- tree(Salary~., hitters)
#Plotting the tree
plot(hitters.tree)
text(hitters.tree, pretty = 0, cex = 0.8)

#Making the error array for LOOCV
error <- rep(1:nrow(hitters))
#test MSE for the LOOCV
set.seed(1)
for(i in 1:nrow(hitters)){
  hitters.tree <- tree(Salary ~ ., data = hitters[-i, ])
  one_predic <- predict(hitters.tree, newdata = hitters[i,])
  error[i] <- (hitters$Salary[i]-one_predic)^2
}
# the mean of error from loocv
mean(error)

#Question 1-(b)
# Making the tree
```

```
hitters.tree <- tree(Salary~., hitters)
# Using LOOCV to determine whether pruning is helpful
hitters_cv<- cv.tree(hitters.tree, K = 263)
# The optimal size for the pruned tree is 9
which.min(hitters_cv$size)
# Making the best pruned tree
prune.hitters<- prune.tree(hitters.tree, best = 9)
plot(prune.hitters)
text(prune.hitters, pretty = 0, cex = 0.8)

#test MSE for the LOOCV
set.seed(1)
for(i in 1:nrow(hitters)){
  hitters.tree <- tree(Salary ~ ., data = hitters[-i, ])
  prune.hitters<- prune.tree(hitters.tree, best = 9)
  one_predic <- predict(prune.hitters, newdata = hitters[i,])
  error[i] <- (hitters$Salary[i]-one_predic)^2
}
# the mean of error from loocv
mean(error)

#Question 1-(c)
# Setting seed as 1
set.seed(1)
# bagging approach with B = 1000
hitters_bag <- randomForest(Salary ~ ., data = hitters,
                           mtry = totpred, ntree = 1000, importance = TRUE)
# To get the result, and the predictors to be the most important
importance(hitters_bag)
varImpPlot(hitters_bag)

# To calculate the test MSE by using LOOCV
error <- rep(1:nrow(hitters))
set.seed(1)
for(i in 1:nrow(hitters)){
  hitters_bag <- randomForest(Salary ~ ., data = hitters[-i, ],
                              mtry = totpred, ntree = 1000, importance = TRUE)
  one_predic <- predict(hitters_bag, newdata = hitters[i,])
  error[i] <- (hitters$Salary[i]-one_predic)^2
}
# the mean of error from loocv
mean(error)
#0.1864304
```

```
#Question 1-(d)
set.seed(1)
# Random forest approach with B = 1000, and m ~~ p/3 --> p = totpred
hitters_rf <- randomForest(Salary ~ ., data = hitters,
                           mtry = round(totpred/3), ntree = 1000, importance = TRUE)
# To get the result, and the predictors to be the most important
importance(hitters_rf)
varImpPlot(hitters_rf)

# To calculate the test MSE by using LOOCV
error <- rep(1:nrow(hitters))
set.seed(1)
for(i in 1:nrow(hitters)){
  hitters_rf <- randomForest(Salary ~ ., data = hitters[-i, ],
                             mtry = round(totpred/3), ntree = 1000, importance = TRUE)
  one_predic <- predict(hitters_rf, newdata = hitters[i,])
  error[i] <- (hitters$Salary[i]-one_predic)^2
}
# the mean of error from loocv
mean(error)

#Question 1-(e)
set.seed(1)
hitters_boost <- gbm(Salary~., data = hitters, distribution = "gaussian",
                     n.trees = 1000, interaction.depth = 1, shrinkage = 0.01)
summary(hitters_boost)
# To calculate the test MSE by using LOOCV
error <- rep(1:nrow(hitters))
set.seed(1)
for(i in 1:nrow(hitters)){
  hitters_boost <- gbm(Salary~., data = hitters[-i, ], distribution = "gaussian",
                       n.trees = 1000, interaction.depth = 1, shrinkage = 0.01)
  one_predic <- predict(hitters_boost, newdata = hitters[i,], n.trees = 1000)
  error[i] <- (hitters$Salary[i]-one_predic)^2
}
# the mean of error from loocv
mean(error)
# test MSE 0.2186296

#Question 2
diabetes <- read.csv("diabetes.csv")
diabetes$Outcome <- as.factor(diabetes$Outcome)

#Question 2-(a)
```

```
#support vector classifier
# set scale = TRUE to standardize the predictors
#Fit a support vector classifier with cost = 10
svmfit <- svm(Outcome ~ ., data = diabetes, kernel = "linear", cost = 10, scale = TRUE)
svmfit$index
summary(svmfit)
svmfit$fitted

# Fit again with cost = 0.1
svmfit <- svm(Outcome ~ ., data = diabetes, kernel = "linear", cost = 0.1, scale = TRUE)
svmfit$index
summary(svmfit)

# Fit a support vector classifier with
# cost parameter 0.001, 0.01, 0.1, 1, 5, 10
set.seed(1)
tune.out <- tune(svm, Outcome ~ ., data = diabetes, kernel = "linear", ranges = list(cost =
c(0.001, 0.01, 0.1, 1, 5, 10)), scale = TRUE)
summary(tune.out)

#To find out the best model
bestmod <- tune.out$best.model
summary(bestmod)
#test error rate
tune.out$best.performance

#Question 2-(b)
# support vector machine with
# polynomial kernel of degree two and cost 0.001, 0.01, 0.1, 1, 5, 10, 100
set.seed(1)
tune.out <- tune(svm, Outcome ~ ., data = diabetes, kernel = "polynomial", degree = 2,scale =
TRUE,
        ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune.out)
# To find out the best model
bestmod <- tune.out$best.model
summary(bestmod)
#test error rate
tune.out$best.performance

#Question 2-(c)
# support vector maching with a radial kernel with
# gamma 0.1 , 0.5, 1, 2, 3, 4, 5, and cost parameter 0.001, 0.01, 0.1, 1, 10, 100
tune.out <- tune(svm, Outcome ~ ., data = diabetes, kernel = "radial",
                ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100), gamma = c(0.1 , 0.5, 1, 2, 3, 4, 5)),
scale = TRUE)
summary(tune.out)
#To find out the best model
bestmod <- tune.out$best.model
summary(bestmod)
#test error rate
tune.out$best.performance
```